TP3 : Corrigé

Exercice 1

1. a) D'après la commande suivante, la chaîne de caractères "baleine" apparaît :

```
grep 'baleine' 20_mille_lieues_sous_les_mers.txt
```

Pour compter le nombre de fois que cette chaîne apparaît, on ajoute l'option -c de grep :

```
grep -c 'baleine' 20_mille_lieues_sous_les_mers.txt
```

et on trouve 81 occurrences.

b) Pour obtenir les occurrences du **mot** baleine, on utilise l'option -w :

```
grep -wc 'baleine' 20_mille_lieues_sous_les_mers.txt
```

On ne trouve plus que 41 occurrences. Il est normal d'en obtenir moins qu'auparavant car dans ce décompte ne figure plus toutes les chaînes qui contiennent le mot baleine.

c) On utilise cette fois l'option -n de grep pour afficher les 41 lignes concernées :

```
grep -nw 'baleine' 20_mille_lieues_sous_les_mers.txt
```

d) On utilise la syntaxe permettant d'exprimer l'union :

```
grep -E -wn 'ciel|terre' 20_mille_lieues_sous_les_mers.txt
```

2. a) La seule difficulté est l'écriture de l'expression régulière étendue dénotant l'ensemble de mots souhaité :

```
grep -E -wc 'le|la|les' 20_mille_lieues_sous_les_mers.txt
```

Vous risquez d'avoir des résultats parasites avec cette recherche ; par exemple la chaîne "le" dans le mot "mêle" pourra être considéré comme étant une occurrence du motif recherché ci-dessus. En effet, les lettres accentuées peuvent ne pas être considérées comme des lettres (selon l'encodage et selon la langue d'installation de la machine - typiquement si cette langue est l'anglais) ce qui perturbe le fonctionnement de l'option -w.

b) On modifie simplement l'expression de la question précédente en :

```
grep -E -wc '[Ll]e|[Ll]a|[Ll]es' 20_mille_lieues_sous_les_mers.txt
```

c) Pour se restreindre aux chaînes en début de ligne, on utilise le méta-caractère ^. Attention au parenthésage :

```
grep -E -wc '^([Ll]e|[Ll]a|[Ll]es)' 20_mille_lieues_sous_les_mers.txt
```

On obtient une commande très similaire pour le décompte en fin de ligne :

```
grep -E -wc '([Ll]e|[Ll]a|[Ll]es)$' 20_mille_lieues_sous_les_mers.txt
```

3. On peut interpréter cette question de plusieurs façons : soit un ligne vide est une ligne ne contenant aucun caractère, soit une ligne vide est une ligne ne contenant aucun caractère visible pour un être humain (auquel cas, on a le droit par exemple d'avoir des espaces ou des tabulations sur la ligne). Dans le premier cas, avec :

```
grep -E -c '^$' 20_mille_lieues_sous_les_mers.txt
```

on trouve 3954 lignes vides et le même résultat peut être obtenu avec

```
grep -E -c '^(\s)*$' 20_mille_lieues_sous_les_mers.txt
```

ce qui montre que le fichier fourni ne contient pas d'espaces parasites.

4. La réponse aux deux sous-questions est oui, comme le montrent les deux commandes suivantes

```
grep 'd.f' 20_mille_lieues_sous_les_mers.txt
grep -E 'd.{3}f' 20_mille_lieues_sous_les_mers.txt
```

5. a) On en trouve 2040 à l'aide de la commande

```
grep -E -c '^[AEIOUYaeiouy]' 20_mille_lieues_sous_les_mers.txt
```

Attention à la casse.

b) La réponse dépend de ce qu'on considère être un signe de ponctuation. Avec la commande ci-dessous, on trouve 9756 lignes qui ne finissent pas par de la ponctuation. Attention à échapper les méta-caractères.

```
grep -E -c '[^\.\?,;!:]$' 20_mille_lieues_sous_les_mers.txt
```

c) Entre le début et la fin de la ligne, il ne doit y avoir que des caractères de l'intervalle A-Z et il doit y en avoir au moins un sans quoi la ligne serait vide :

```
grep -E -n '^[A-Z]+$' 20_mille_lieues_sous_les_mers.txt
```

Seules les lignes 4274 et 12072 répondent à ce critère.

6. a) Il y a des chiffres dans le document comme le montre :

```
grep -E '[0-9]' 20_mille_lieues_sous_les_mers.txt
```

b) Attention, ici on demande les nombres, il faut donc un peu modifier l'expression régulière pour autoriser les successions de chiffres. Une fois détectés, on liste toutes les occurrences de nombres avec -o puis on dédoublonne avec sort --unique :

```
grep -E -o '[0-9]+' 20_mille_lieues_sous_les_mers.txt |sort --unique
```

Remarque : L'analyse de ces nombres donne une bonne idée de l'époque à laquelle le récit a lieu.

7. a) Le principe est le même que dans la question précédente : on détermine une expression régulière adpatée, on liste les occurrences dans le texte correspondant à ce motif puis on les dédoublonne :

```
grep -E -wo '[a-z]*ar?bo[a-z]*' 20_mille_lieues_sous_les_mers.txt | sort --unique
```

Remarquons qu'avec cette commande, on ne détectera pas les mots contenant "abo" avec un "r" éventuel entre le "a" et le "b" qui commencent par une majuscule. On pourrait modifier l'expression régulière de sorte à les inclure.

b) Cette commande détecte les suites de 15 caractères compris entre deux espaces et non interrompues par un espace. Parmi elles, on a les mots de 15 lettres mais aussi des mots composés (avec un tiret), des suites comprenant une apostrophe ou des suites finissant par un signe de ponctuation. On cherche à les éliminer dans la question suivante.

c) Pour ce faire, on oblige chacun des 15 caractères à être une lettre (éventuellement en majuscule) :

```
grep -E -wo '[a-zA-Z]{15}' 20_mille_lieues_sous_les_mers.txt |sort --unique
```

8. a) L'énoncé nous précise le motif qu'il nous faut rechercher :

```
grep -E -o '_[^_]*_' 20_mille_lieues_sous_les_mers.txt |sort --unique
```

On interdit aux caractères situés entre deux underscore d'être un underscore, sans quoi dès que la chaîne de caractères "_bidule_ et _machin_" se trouve sur une même ligne, elle serait reconnue comme satisfaisant les contraintes de l'expression régulière.

b) On peut commencer par proposer ceci :

```
grep -E '[XIV]+\.' 20_mille_lieues_sous_les_mers.txt
```

On obtient un résultat proche de celui attendu mais perturbé de quelques résultats parasites qu'on peut chercher à épurer en raffinant l'expression régulière. Par exemple avec

```
grep -E '[XIV]+\.(\s)+[A-Z]' 20_mille_lieues_sous_les_mers.txt
```

on n'obtient plus qu'un seul résultat parasite, qu'il est en fait impossible de supprimer car le motif qui y est détecté est indissociable du motif permettant de repérer un chapitre.

c) En parcourant le fichier fourni, on remarque que les illustrations sont signalées par des chaînes de la forme "[Illustration: <description de l'illustration>], d'où :

```
grep -E '\[Illustration:.*\]' 20_mille_lieues_sous_les_mers.txt
```

d) En parcourant le fichier fourni, on remarque que les notes en bas de page sont signalées par des nombres entre crochets d'où :

```
grep -E '\[[0-9]+\]' 20_mille_lieues_sous_les_mers.txt
```

9. Plusieurs stratégies sont possibles. On propose la suivante : les noms des personnages ont une majuscule et peuvent se trouver au milieu d'une phrase, contrairement aux noms communs : on va donc rechercher les mots qui commencent par une majuscule et ne sont pas en début de phrase. Ensuite, on comptera le nombre d'occurrences de chaque mot ainsi trouvé et ceux qui apparaîtront le plus seront les personnages principaux. Cette tactique a ses limites :

   - On détectera tous les noms propres, pas seulement les noms des personnes.
   - Si le nom d'un protoganiste est systématiquement en début de phrase, il ne sera pas détecté.
   - Cette stratégie part du principe qu'un personnage principal verra souvent son nom cité, ce qui n'est pas garanti (par exemple si le roman est narré à la première personne).

   La commande

   ```
   grep -E -ow '[a-z]+(\s)[A-Z][a-z]+' 20_mille_lieues_sous_les_mers.txt
   ```

   permet de déctecter les occurrences de mots qui commencent par une majuscule en milieu de phrase mais il faut encore les purger du mot qui les précède. Pour ce faire on redirige la sortie de ce premier grep sur l'entrée d'un nouveau grep qui détecte les mots après l'unique espace de chacune des occurrences précédentes :

   ```
   grep -E -ow '[a-z]+(\s)[A-Z][a-z]+' 20_mille_lieues_sous_les_mers.txt |
       grep -E -o '(\s).*'
   ```

   Il ne reste plus qu'à compter le nombre de fois que chaque chaîne apparaît :

   ```
   grep -E -ow '[a-z]+(\s)[A-Z][a-z]+' 20_mille_lieues_sous_les_mers.txt |
       grep -E -o '(\s).*' | sort | uniq -c
   ```

   Les noms apparaissant plus de 50 fois sont : Aronnax (62 fois), Canadien (193 fois), Conseil (310 fois), Ned (259 fois), Nemo (451 fois) ce qui est une approximation des personnages principaux très honorable pour quiconque a lu le roman étudié dans cette partie.

Exercice 2

Cet exercice ne devrait poser aucune difficulté !

1. `ls /usr/bin/ | grep 'grep'`

   On remarque que la commande grep admet de nombreuses variantes.

2. `ls /usr/bin/ | grep -E 'a$'`

3. `ls /usr/bin/ | grep -E '^.{2}$'`

   On retrouve dans le résultat de cette commande la commande ls.

4. `ls /usr/bin/ | grep -E '.*r.*r.*'`

Partie 3

Après complétion on obtient :

T	O	B	E	O
R	N	O	T	T
O	B	E	A	R
E	G	U	L	A
R	R	E	G	X